Objects Over Sugar as Closure

By: Keanu Mendez, Austin Marlett, Norbert Krajewski, Payne Persons, and Jeremy Goff

The Big Idea

Objects are NOT real – they are just functions in disguise.

- Objects = functions + closures (via desugaring)
- The interpreter never sees objects – we just translate "objects" into simpler core code
- We are not implementing objects, we are compiling them
- The interpreter only sees functions, not objects.

Language Pipeline

Our language pipeline follows this set of processes:

Code → Parser → Desugarer → Interpreter → Result

- **Parser:** Converts code into an AST
- **Desugarer:** Translates objects into functions
- **Interpreter:** Runs only the core language

The key part of our pipeline is the **desugarer**, which translates our objects into functions so that the interpreter only runs the core language.

Objects → Functions → Execution

What We Created

We introduced new syntax which is translated into the simpler core language.

New Syntax:

- **Object Creation**

```
{object
   {field x value}
   {method name {lam arg body}}}
```

- **Method Calling**

```
{send obj "method" arg}
```

- **Class Creation**

```
{class {params}
   {field ...}
      {method ...}}
```

All of the new syntax features are syntactic sugar that are translated into functions before execution.

Example of Desugaring

Before (Object Syntax)

```
{object
 {field a 1}
 {method get-a {lam _ a}}}
```

After (Desugared into Functions)

```
(let1E 'a (numE 1)
       (lamE 'msg
             (ifE (equalE (varE 'msg) (stringE "get-a"))
                  (lamE '_ (varE 'a))
                  (numE 0))))
```

- The object becomes a function (dispatcher)
- Methods become functions selected based on a message
- Fields are stored using closures

Method Call

Before (Surface

```
{send o "get-a" 0}}
```

After (Desugared)

```
(appE
 (appE (varE 'o) (stringE "get-a"))
 (numE 0))
```

- We send a message to our object. (In this case, "get-a" to "o")
- This will return the method (a function).
- Finally, we apply the method to the argument (0).

Message → Function → Resul

Classes

- The only difference between objects and classes is that classes have parameters.
- Parameters are just extra variables defined before an object.
- So, classes are sugared objects; we only must prepend the object with the variable definitions

```
{class {p1}
 {field a p1}
 {method get-a {lam _ a}}}
```

```
(lamE 'p1
      (let1E 'a (varE 'p1)
             (lamE 'msg
                   (ifE
                    (equalE (varE 'msg)
                            (stringE "get-a"))
                    (lamE '_ (varE 'a))
                    (numE 0)))))
```

Recap

Objects are just functions (via desugaring).

- Object = functions that take messages and decides what to do (dispatcher)
- Fields = variables in closures
- Methods = functions returned when a message is sent
- Classes = functions that create objects with different parameters

Everything we built - like objects, fields, and methods - are just functions created with desugaring.

The interpreter never sees objects – only functions.

Now, For Some In-Class Examples.